使用 UB-tree 索引时态 XML*)

赵 林 王新军
(山东大学计算机科学与技术学院 济南 250061)

摘 要 如何在 XML 文档中表达时间相关的数据,跟踪历史信息和恢复文档在以前任意时刻的状态的问题,在最近的研究中受到不少的关注。许多文献提出了各种不同的模型。我们将这一类的问题归为索引时态 XML 文档的问题。本文将时态 XML 文档转换到 n 维空间的节点和直线,使用 UB-tree 对这些 n 维空间的节点和直线进行索引,并针对时态查询提出了新的查询算法。通过实验证明,这样的索引比之前针对时态模型提出的索引具有更好的性能。

关键词 时态 XML, UB-树,n-维空间,TXPath 查询表达式

Indexing Temporal XML Using UB-tree

ZHAO Lin WANG Xin-Jun
(School of Computer Science and Technology, Shandong University, Jinan 250061)

Abstract Different models have been proposed recently for representing temporal data, tracking historical information, and recovering the state of the document as of any given time, in XML documents. We address the problem of indexing temporal XML documents. In this paper ,we transform the temporal XML data into points and lines in the n- dimensional space and use the UB-tree to index them. A new algorithm for temporal query has been proposed based on the UB-tree index. Though the experiments we can see that this index has a better performance than the index proposed before.

Keywords Temporal XML, UB-tree, n- dimensional space, TXPath query expression

1 介绍

XML 是 Web 上进行信息表示与交换的一个层次数据格式。随着大量 XML 数据的出现,如何有效地索引,存储和查询这些 XML 数据越来越受到人们的关注。众所周知,在 Web 和一些商业需求中,时间信息是反映现实世界实体状态的一个重要部分。因此提高时态 XML 的查询速度具有重要的意义。文[1]提出了时态 XML 的查询语言 TXPath,如 COMPANY / employee/ score [@from≥05 and @to≤06],表示找出员工中,2005 年到 2006 年在校读书的人的成绩。

```
〈COMPANY〉
  〈employee[96,now]〉Allen〈score [91,95]〉91〈/score 〉〈/employee
〉
  〈employee [00,06]〉Jack〈score [01,05]〉82〈/score 〉〈/employee 〉
  〈employee [05,now]〉Steve〈score [02,06]〉86〈/score 〉〈/employee
〉
〈/COMPANY 〉
```

图 1 时态 XML 文档片段举例

图 1 是一个公司员工信息的时态 XML 文档片段。其中与 employee 关联的时间表示其在公司工作的时间,与 score 相关的是该员工在大学读书的时间,其值为成绩,例如员工 Allen,从 96 至今在公司工作,大学期间(91 年到 95 年)成绩为 91。注意 Jack 是从上大学前就在公司工作的员工,因此他的两个时间是一种包含关系;Steve 是大学期间开始在公司工作的。

最近的研究中,许多文献提出了各种不同的索引模型。其中大多数都是在时态 XML 文档更新时产生一个新版本,如版本索引(version index)[3]。这种思想在时态数据库中曾使用过,针对时间的处理有大量的解决方法[2]。但是这样造成了查询时需要遍历不同版本的数据,降低了查询处理的性能。有些研究则借助非时态 XML 索引的思想,在其上增加对时间处理的算法。例如 TempIndex[4]采用了普通 XML 索引 1-index [9]的思想,但对带有谓词的路径查询,则需要遍历文档树的大部分结点。这就不能很好处理含有大量数据的 XML 文档。此外文[7,8,10,11]提出的索引主要是通过结构摘要类[5]的思想来简化路径信息。

上面索引中,对时间的处理都是以几个离散的时间点作为索引关键字。随着文档中时间信息的增加,这种方法并不能比较均匀地索引时间信息,而且往往会存在冗余。时间应该是一个连续的概念,用坐标系中的一个坐标轴表示更能反映出其连续性。

本文将时态 XML 文档与多维空间的点或者直线进行映射,同时用空间中一个专门的坐标轴表示文档的时间信息。UB-tree(Universal B-tree)是在多维空间中快速定位节点的索引技术[4,6,12]。本文使用 UB-tree 对文档转换后的节点和直线进行索引,避免了之前索引中的遍历多个版本和遍历整个文档的问题。实验表明,由于避免了大量不必要的遍历,这种索引有比较好的性能。

本文第 2 部分介绍相关的概念。第 3 部分给出 UB-tree 索引时态 XML 文档的方法。第 4 部分讨论在这个索引下的查询处理过程。第 5 部分给出我们的实验数据。最后得出结

*)项目基金:教育部科学技术研究重点项目(03102)。赵 林 硕士研究生,主要研究方向:XML 索引技术,数据仓库;王新军 博士,教授,主要研究方向:面向对象数据库,XML 索引技术,数据仓库,算法分析与设计,网络集成和管理。

论。

2 相关概念

定义 1 假定 Ω 表示 n 维空间。对于空间中的一点 $\phi\in\Omega$，将其 n 个坐标值用二进制表示为 $A_i=A_{i,s-1}A_{i,s-2}\cdots A_{i,0}$，$1\leqslant i\leqslant n$，则其 Z-地址 $Z(\phi)$ 的定义为：

$$Z(\Phi)=\sum_{j=0}^{s-1}\sum_{i=1}^{n}A_{ij}\,2^{nj+i-1}$$

利用 Z-地址的方法，我们可以将 n 维空间与一维空间建立对应关系，并且可以根据其 Z-地址对 n 维空间节点进行排序。

定义 2 Z-区域 $[\alpha:\beta]$ 是指包含 Z-地址在 $[\alpha,\beta]$ 的节点的区域。

UB-tree 的结构类似于 B+树。B+树中只有叶子节点保存有关键字信息。而 UB-tree 中的关键字使用的就是 Z-区域。根据 Z-区域将空间节点保存在相应的页面中，可以使得经常使用的节点存储在相邻的位置，这对于空间中的区域查询处理有比较好的性能。

定义 3（路径内容） 给定 XML 查询表达式 e1/e2/…/ek，其路径内容为 s1/s2/…/sk，其中 si 为标签 ei 对应的值。

例如，在图 1 中，employee / score 对应的路径内容为 Allen/91，Jack/82，Steve /86。

3 UB-tree 索引时态 XML 文档

下面我们给出时态 XML 文档到 n 维空间节点的转换方法。这里考虑的路径为从根节点开始到叶节点结束的路径内容，可以包含任意个的时间约束条件。n 维空间中的第 n 维坐标为时间坐标。

算法 1

输入：时态 XML 文档中的一个路径内容
输出：该路径内容在 n 维空间中的表示
步骤：

1. 引入函数 sig(x)，将字符串 x 转换成固定长度的唯一的二进制编码表示。对每一个路径内容分配 pcd，并在转换后路径对应的节点和直线上关联这个 pcd。
2. 对不包含时间信息的路径内容转向第 3 步。对包含时间信息的路径内容转向第 4 步。
3. 不包含时间信息的路径内容，其形式表示为 s1/s2/s3/…/sk。其对应的 n 维空间节点的坐标为(sig(s1)，sig(s2)，sig(s3)…sig(sk)…0)。若 si 为空，则 sig(si)=0。若 $n-1>k$，则对于每一个 $j,k<j<n$，第 j 个坐标为 0。算法结束。
4. 包含时间信息的路径内容，其形式可以表示为 s1/s2/…/si [T1]/…/sp[T2]/…/sj [Tm]/…/sk。其中 T1，T2…Tm 表示 m 个时间约束条件。在 n 维空间中用 1 个节点，$m-1$ 条直线表示：
   s1/s2/…/si−1，其对应的节点坐标为 (sig(s1)，sig(s2)…sig(si−1)，0…0)
   s1/s2/…/si−1/si [T1]/…/sp−1，其对应的坐标为(sig(s1)，sig(s2)…sig(si−1)，sig(si)…sig(sp−1)，0…0，T1)，这是 n 维空间的一条直线。因为 T1 是一个时间区间
   s1/s2/…/si [T1]/…/sp [T2]/…，其对应的坐标为(sig(s1)，sig(s2)…sig(si)…sig(sp)，0…0，T2)，这是 n 维空间中的一条直线
   ……
   s1/s2/…/si [T1]/…/sp [T2]/…/sj [Tm]/…/sk，其对应的节点坐标维为(sig(s1)，sig(s2)…sig(si)…sig(sp)…sig(sj)…sig(sk)…Tm)。若 si 为空，则 sig(si)=0。若 $n-1>k$，则对于每一个 $j,k<j<n$，第 j 个坐标为 0。算法结束。

对于 n 维空间中的直线我们可以得到其对应节点的 Z-地址范围，从而可以使用 Z-区域来表达直线。这样我们就可以将时态 XML 文档对应的 n 维空间节点和直线插入到 UB-tree 的索引中进行索引。

我们将图 1 的文档转换为 n 维空间节点、直线。文档中最长的路径长度为 3，由于时间要用专门的一维表示，因此在 4 维空间中就可以表示这个文档。为了讨论的方便，我们暂时不考虑文档中的根节点，这样在 3 维空间中就可以给出相应的表示结构。

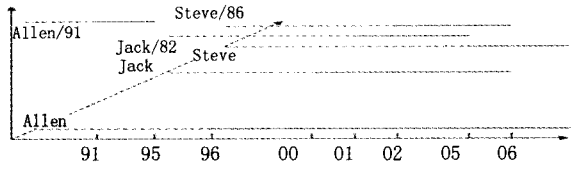

图 2 图 1 对应的三维空间表示

根据算法 1，图 1 中有时间约束的 3 个路径内容转换为三维空间的 6 条直线。例如 Allen/91 对应图 2 中的两条直线，因为其上有两个时间约束：[96，now]和[91，95]。这些直线对应的 Z-范围将被插入到 UR-tree 中。

4 查询处理

对于时态 XML 上的查询，我们可以转换成对坐标轴上的坐标的处理。根据查询表达式中的约束条件，对应的结果往往在一个满足坐标限制的空间内，我们称之为查询空间。我们可以根据查询空间对应的 Z-区域，在 UB-tree 中找到满足条件的多维空间节点或者直线，从而找到其对应的文档元素的值。我们给出使用 UB-tree 索引时态 XML 文档的查询算法。

算法 2

输入：查询表达式 R1/R2/…/Ri1 [T1]/…/Ri2 [T2]/…/Rim [Tm]/…/Rn，其中 T1…Tm 为 m 个时间谓词，非时间查询条件未显式标出。
输出：满足查询要求的文档中的路径内容。
步骤：

1. 若查询表达式中不包含时间谓词，则转向步骤 2，否则转向步骤 3。
2. 若查询表达式不包含任何时间谓词，那么我们在时间轴为 0 的 n−1 维空间上，根据非时间谓词构造查询空间，找到在时间轴为 0 空间中的节点，就是要求的路径内容。算法结束。
3. 根据查询表达式，生成 m 个查询空间。R1/R2/…/Ri1 [T1] 对应查询空间的 $\Omega1$ 由其中的非时间谓词和时间范围 T1 决定。R1/R2/…/Ri1/…/Ri2 [T2]对应的查询空间 $\Omega2$ 由其中的非时间谓词和时间范围 T2 决定。
   ……
   R1/R2/…/Ri1/…/Ri2/…/Rim [Tm]/…/Rn 为第 m 个查询空间 Ωm，由其中的非时间谓词和时间范围 Tm 决定。
4. 对于穿过空间 Ωj(R1/R2/…/Rij [Tj])的直线，寻找其在时间轴为 0 的空间上的投影点坐标是(sig(R1)，sig(R2)…sig(Rij)，0…0)的形式的。记录这条直线对应的 pcd。
5. 将每一个查询空间，例如 Ωj(R1/R2/…/Rij [Tj])，向时间轴为 0 的空间上投影。看投影区域是否包含坐标形式为(sig(R1)，sig(R2)…sig(Rij)，0…0)的节点，记录节点对应的 pcd。
6. 将步骤 4 和步骤 5 中各个查询空间包含的 pcd 集合求交集。交集中的 pcd 就是满足查询表达式的时态 XML 文档中的路径内容。算法结束。

我们在算法中的步骤 2 和步骤 3 使用已经建立好的 UB-tree 索引。由于算法中查询空间的范围实际上是一个 Z-区域，这样就可以通过在 UB-tree 中找到这个 Z-区域，并对其中包含的节点或者直线进行判断。

例如：COMPANY/Employee[@from≥97 and @to≤98 and /score[@from≥92 and @to≤93]]，查找 97 年到 98 年在公司工作的，且 92 年到 93 年在校读书的员工。

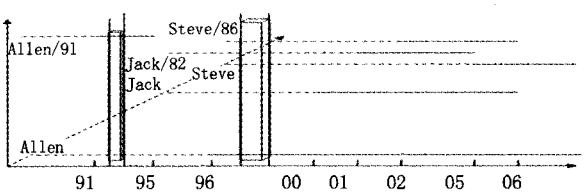

图 3 查询表达式对应的查询空间

根据算法，由于查询中存在两个时间谓词，则产生两个查询空间，图 3 用立方体表示。穿过 92 到 93 的立方体的直线，

（下转第 233 页）

结束语 本文提出了一种求最简规则的方法。该方法的一个优点是将差别矩阵扩展后，可以实现分布式的规则获取。该方法的另一个优点是能够求得多个最简规则，当最终结果的 $RL[d]$ 中存在多个长度相同的合取式时，可求得多个可选的最简规则。

参 考 文 献

1 Skowron A, Rauszer C. The discernibility matrices and functions in information systems [A]. In: Slowinski I. Intelligent decision support- handbook of applications and advances of the Rough Sets theory [C]. Dordrecht: Kluwer Academic Publisher, 1991. 331～362
2 Pawlak Z. Rough Set approach to multi- attribute decision analysis [J]. European Journal of Operational Research, 1994, 72: 443～459
3 王国胤. Rough 集理论与知识获取[M]. 西安：西安交通大学出版社，2001
4 韩祯祥，张琦，文福拴. 粗糙集理论及其应用综述. 控制理论与应用[J]. 1999, 16(2): 153～165
5 Hassanien A E, et al. Rough Set Approach for Generation of Classification Rules of Breast Cancer Data. Institute of Mathematics and Informatics, 2004, 15(1): 23～38
6 Pawlak Z. Rough sets and intelligent data analysis [J]. Information Science, 2002(147): 1～12
7 叶东毅，陈昭炯. 不相容决策表属性约简计算的一个可辨识矩阵方法[J]. 福州大学学报(自然科学版)，2005, 33(1)
8 常黎云，王国胤，吴渝. 一种基于 Rough Set 理论的属性约简及规则提取方法. 软件学报，1999, 10(11)

(上接第 72 页)

其在时间为 0 的平面内的投影点坐标为(sig(Allen),0,0)，满足算法的要求。这个查询空间在时间为 0 的平面的投影中不包含满足(sig(Allen),0,0)的节点。这样为该查询空间记录这条直线的 pcd，为 1。对于另一个立方体，穿过的直线在时间为 0 的平面内的投影点坐标为(sig(Allen),91,0)，满足算法要求，记录其 pcd，同样为 1。两个空间的 pcd 的交集就是{1}。因此满足要求的路径内容为 pcd 为 1 的 Allen/91。由于要求的为 employee 对应的值，因此结果为 Allen。

5 实验

我们的实验环境的 CPU 为 Pentium4 3.00GHz，1GB 内存，120G IDE 硬盘，操作系统为 Microsoft XP。使用 Microsoft Visual C++ 6.0 进行实验。实验采用的数据集是包含更多员工数据的 XML 文件，大小 30MB。结点数目为 1022976。采用文[4]的索引 TempIndex 作为比较。同时比较使用 DOM 解析的时间，这是一种对无索引情况下查询的比较。针对 XML 文件建立的 UB-tree 索引大小为 1.5k，TempIndex 索引中结点数目为 2732。

对于查询：COMPANY/employee[@from≥97 and @to≤98 and /score[@from≥92 and @to≤93]]。其性能比较的柱状图如下。

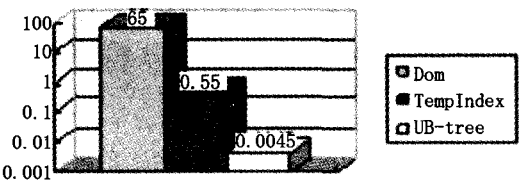

图 4 时态查询的性能比较

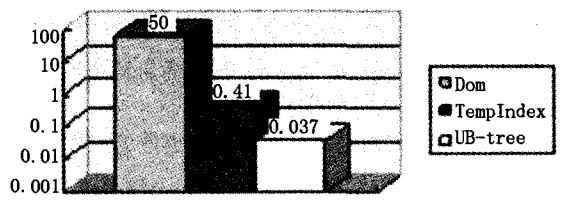

图 5 非时态查询的性能比较

从图 4 中可以看出，这三种方法的处理性能有着较大的差别。其中 DOM 思想，只能基于整个文档的遍历，这样当处理数据量大的 XML 的时候，其性能往往较差，因为存在大量无谓的数据遍历和磁盘访问的问题。TempIndex[4] 要直接操作一定量文档节点。而使用 UB-tree 处理这样的查询，只需根据查询表达式产生查询空间。由于 UB-tree 高度较小，在 UB-tree 的结构中可以很快地定位到查询空间对应的 Z-区域，从而得到对应的文档内容的表示，因此有着比较好的性能。

对于非时间谓词的查询 COMPANY/Employee[/score>91]]。图 5 给出了其性能的比较。DOM 方法依然是遍历整个文档。而 TempIndex 的查询消耗时间有所降低，因为其借助 1-index 的思想，针对简化了的 XML 文档图进行操作，然而对于谓词还是要进行局部的文档遍历。而 UB-tree 依然有着较好的性能，并且消耗时间较上一个查询有所减少。原因在于 UB-tree 没有进行大量的文档遍历，同时由于没有时间谓词，查询空间减小，只针对于时间轴为 0 的空间，查询处理的复杂度降低，所以消耗的时间更少。

总结 本文提出了在时态 XML 文档下的一种新的索引思想，使用 UB-tree 进行索引。基本思想是，将时态 XML 文档转换成多维空间的节点或者是直线。利用 UB-tree 对其进行索引，并给出了针对时态查询表达式的查询算法。由于 UB-tree 有着比较小的高度，因此能够较快地定位到要求的节点。经过理论和实验数据的分析，可以看出这种方式比之前文献介绍的 TempIndex 能够在一定程度上减少不必要的路径遍历和磁盘访问次数，因此有更好的性能。下一步的工作，在 XML 文档到 n 维空间转换过程中，用到了多条直线。改进转换算法，简化这种结构，可以有效地提高我们的查询效率。

参 考 文 献

1 Vaisman A A, Mendelzon A O, Molinari E, Tome P. Temporal XML: Model, Language and Implementation
2 Salzberg B, Tsotras V J. Comparison of Access Methods for Time-Evolving Data. ACM Computing Surveys, 1999, 31(2)
3 Dyreson C E. Observing transaction-time semantics with TTX-Path. In WISE, 2006. 193～202
4 Mendelzon A O, Rizzolo F, Vaisman A. Indexing Temporal XML Documents. In: Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004
5 孔令波，唐世渭，杨冬青，王腾蛟，高军. XML 数据索引技术. 软件学报，2005, 16(12)
6 Bayer R. The Universal B-Tree for multidimensional indexing: General Concepts. In: Proc. of World-Wide Computing and its Applicazions 97 (WWCA 97). Tsukuba, Japan, 1997
7 Kaushik R, Bohannon P, Naughton J F, Korth H. Covering indexes for branching path queries. In ACM SIGMOD, Wisconsin, Madison, 2002. 133～144
8 Goldman R, Widom J J. Dataguides: Enabling query formulation and optimization in semistructured databases. In VLDB, Athens, Greece, 1997. 436～445
9 Milo T, Suciu D. Index structures for path expressions. In: Beeri C, Buneman P, eds. Proc. of the 1999 Int'l Conf. on Database Theory (ICDT). LNCS 1540, Jerusalem: Springer-Verlag, 1999. 277～295
10 Cooper B, Sample N, Franklin M J, Hjaltason G R, Shadmon M. A fast index for semistructured data. In VLDB, Rome, Italy, 2001. 341～350
11 Li Q, Moon B. Indexing and querying XML data for regular path expressions. In VLDB, Rome, Italy, 2001. 361～370
12 Markl V. Mistral: Processing Relational Queries using a Multidimensional Access Technique, http://mistral.in.tum.de/results/publications/Mar99.pdf, 1999